(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 280 068 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.01.2003 Bulletin 2003/05

(51) Int Cl.7: G06F 17/27

(21) Application number: 02014717.9

(22) Date of filing: 03.07.2002

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 16.07.2001 US 906467

(71) Applicant: MICROSOFT CORPORATION
Redmond, WA 98052 (US)

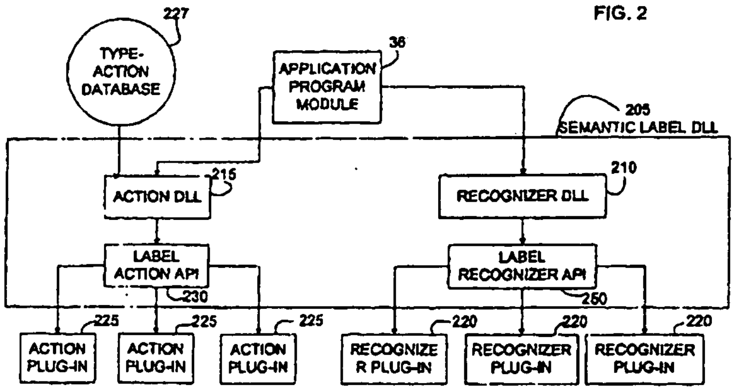
(72) Inventors:
- Reynar, Jeff
Woodinville, Washington 98072 (US)
- Wang, Ziyi
Redmond, Washington 98052 (US)
- Wolff, Roger
Redmond, Washington 98052 (US)
- Huynh, Tuan
Seattle, Washington 98121 (US)
- Higashiyama, Nobuya
Issaquah, Washington 98029 (US)
- Ammerlaan, Michael
Sammamish, Washington 98053 (US)

(74) Representative: Grünecker, Kinkeldey, Stockmair & Schwanhäusser Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) Application program interfaces for semantically labeling strings and providing actions based on semantically labeled strings

(57) Application program interfaces (API) are provided for labeling strings while a user is creating a document and providing user actions based on the type of semantic label applied to the string. A recognizer API is provided and includes properties and methods or instructions which allow recognizer plug-ins to semantically label strings of text or cells or information. An action API is provided and includes properties and methods that are called upon when a user initiates particular actions such as opening a web browser, going to a particular URL, or opening an instance of a word processing or spreadsheet program. After the strings are annotated with a type label, application program modules may use the type label to provide users with a choice of actions. If the user's computer does not have any actions associated with a type label, the user may be provided with the option to surf to a download Uniform Resource Locator (URL) and download action plug-ins for that type label. One or more recognizer plug-ins perform the recognition of particular strings in an electronic document. The recognizer plug-ins may be packaged with an application program module or they may be written by third parties to recognize particular strings that are of interest. One or more action plug-ins provide possible actions to be presented to the user based upon the type label associated with the string.

FIG. 2



Description

Technical Field

[0001] This invention relates to application program interfaces for semantically labeling strings of text during creation of an electronic document and providing a selection of actions that may be performed based on the semantically labeled strings.

Background of the Invention

[0002] Electronic documents such as word processing documents and spreadsheet documents typically include semantic information that would be helpful if the information was recognized as such. Recognition and use of this semantic information could result in increased interoperability between desktop software applications and other desktop applications and/or web-based applications.
[0003] The ability to recognize strings of text, such as in search engines, is well-known. Additionally, various information retrieval systems have capabilities to label documents. For example, the LEXIS-NEXIS service provides links in some of its documents for company names, famous people and stock ticker symbols.
[0004] However, none of the services described above allow strings of text to be labeled with semantic information on-the-fly, i.e., as a user is typing text into a document and creating a document. Thus, there is a need for a method and system for semantically labeling strings while a user is creating a document and providing user actions based on the type of semantic label applied to the string. There is further a need for application program interfaces (API) for labeling strings while a user is creating a document and providing user actions based on the type of semantic label applied to the string.
[0005] It is with respect to these considerations and others that the present invention has been made.

Summary of the Invention

[0006] The present invention provides application program interfaces (API) for labeling strings while a user is creating a document and providing user actions based on the type of semantic label applied to the string. A recognizer API is provided and includes properties and methods or instructions which allow recognizer plug-ins of a recognizer dynamic-link library (DLL) to semantically label strings of text or cells or information. An action API is provided and includes properties and methods that are called upon when a user initiates particular actions such as opening a web browser, going to a particular URL, or opening an instance of a word processing or spreadsheet program.
[0007] These and other features, advantages, and aspects of the present invention may be more clearly understood and appreciated from a review of the following detailed description of the disclosed embodiments and by reference to the appended drawings and claims.

Brief Description of the Drawings

[0008]

Fig. 1 is a block diagram of a computer that provides the exemplary operating environment for the present invention.
Fig. 2 is a block diagram illustrating an exemplary architecture for use in conjunction with an embodiment of the present invention.
Fig. 3 is a flow chart illustrating a method for semantically labeling strings during creation of an electronic document.
Fig. 4 is an illustration of a display of a semantic category and its associated dropdown menu.
Fig. 5 is a block diagram illustrating properties and methods associated with an action API and a recognizer API.

Detailed Description

[0009] The present invention is directed toward application program interfaces (API) for labeling strings while a user is creating a document and providing user actions based on the type of semantic label applied to the string. A recognizer API is provided and includes properties and methods or instructions which allow recognizer plug-ins of a recognizer dynamic-link library (DLL) to semantically label strings of text or cells or information. An action API is provided and includes properties and methods that are called upon when a user initiates particular actions such as opening a web browser, going to a particular URL, or opening an instance of a word processing or spreadsheet program. A string is defined as a data structure composed of a sequence of characters usually representing human-readable text.
[0010] After the strings are annotated with a type label, application program modules may use the type label to

provide users with a choice of actions. If the user's computer does not have any actions associated with a type label, the user may be provided with the option to surf to a download Uniform Resource Locator (URL) and download action plug-ins for that type label. One or more recognizer plug-ins perform the recognition of particular strings in an electronic document. The recognizer plug-ins may be packaged with an application program module or they may be written by third parties to recognize particular strings that are of interest. One or more action plug-ins provide possible actions to be presented to the user based upon the type label associated with the string.

[0011] In one embodiment, the invention is incorporated into a suite of application programs referred to as "OFFICE", and more particularly is incorporated into a preferred word processing application program entitled "WORD 10.0", a preferred spreadsheet application program entitled "EXCEL 10.0", a preferred e-mail application program entitled "OUTLOOK 10.0" and a preferred web browser application program entitled "INTERNET EXPLORER 6", all marketed by Microsoft Corporation of Redmond, Washington. Briefly described, the preferred application programs allow a user to create and edit electronic documents by entering characters, symbols, graphical objects, and commands.

[0012] Strings are recognized and annotated, or labeled, with a type label. After the strings are annotated with a type label, application program modules may use the type label and other metadata to provide users with a choice of electronic commerce actions. If the user's computer does not have any actions associated with that type label, the user may be provided with the option to surf to a download Uniform Resource Locator (URL) and download action plug-ins for that type label.

[0013] Having briefly described an embodiment of the present invention, an exemplary operating environment for the present invention is described below.

Exemplary Operating Environment

[0014] Referring now to the drawings, in which like numerals represent like elements throughout the several figures, aspects of the present invention and the exemplary operating environment will be described.

[0015] Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of an application program that runs on an operating system in conjunction with a personal computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, cell phones, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0016] With reference to Fig. 1, an exemplary system for implementing the invention includes a conventional personal computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples the system memory to the processing unit 21. The system memory 22 includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 further includes a hard disk drive 27, a magnetic disk drive 28, e.g., to read from or write to a removable disk 29, and an optical disk drive 30, e.g., for reading a CD-ROM disk 31 or to read from or write to other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage for the personal computer 20. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM disk, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

[0017] A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more application programs 36, a word processor program module 37 (or other type of program module), program data 38, and other program modules (not shown).

[0018] A user may enter commands and information into the personal computer 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers.

[0019] The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer 20, although only a memory storage device 50 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, Intranets and the Internet.

[0020] When used in a LAN networking environment, the personal computer 20 is connected to the LAN 51 through a network interface 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the WAN 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0021] Fig. 2 is a block diagram illustrating an exemplary architecture 200 for use in conjunction with an embodiment of the present invention. The architecture includes an application program module 36, such as word processor program module 37 (Fig. 1). The application program module 36 is able to communicate with a recognizer dynamic-link library 210 (hereinafter recognizer DLL) and an action dynamic-link library 215 (hereinafter action DLL) as a user is creating or editing an electronic document. According to a preferred embodiment, the recognizer DLL 210 and the action DLL 215 are combined in a semantic label DLL 205. The recognizer DLL 210 controls a number of recognizer plug-ins 220. The action DLL 215 controls a number of action plug-ins 225. The action DLL also controls a type-action database 227.

[0022] In a preferred embodiment, the action plug-ins and recognizer plug-ins are Automation Servers. Automation Servers are well-known software components which are assembled into programs or add functionality to existing programs running on the Microsoft WINDOWS® operating system. Automation Servers may be written in a variety of computing languages and may be un-plugged from a program at run time without having to recompile the program. It should also be understood that, in a preferred embodiment, the action DLL and recognizer DLL are merged into a single DLL.

[0023] The recognizer DLL 210 handles the distribution of strings from the electronic document running on the application program module 36 to the individual recognizer plug-ins 220. The recognizer plug-ins 220 recognize particular strings in an electronic document, such as a word processing document, a spreadsheet document, a web page, etc. The recognizer plug-ins 220 may be packaged with the application program module 36 or they may be written by third parties to recognize particular strings that are of interest. Typically, the recognizer DLL 210 passes strings to the recognizer plug-ins 220 in one paragraph or cell value increments.

[0024] As part of recognizing certain strings as including semantic information, the recognizer plug-ins 220 determine which strings are to be labeled and how they are to be labeled. After receiving these results from the various recognizer plug-ins 220, the recognizer DLL 210 sends semantic categories to the application program module. In a preferred embodiment, a semantic category comprises the recognized string, a type label, and a download URL. A semantic category may also comprise metadata. The recognizer plug-ins 220 each run separately and the recognizer DLL 210 is responsible for handling the asynchronicity that results from different recognizer plug-ins returning results with different delays.

[0025] After a string is labeled by a recognizer plug-in 220 and a semantic category is sent to the application program module 36, the user of the application program module 36 will be able to execute actions that are associated with the type label of the semantic category. The action DLL 215 manages the action plug-ins 225 that are run to execute the actions. As with the recognizer plug-ins 220, the action plug-ins 225 may be packaged with the application program module 36 or written by third parties to perform particular actions that are of interest to the third party. The action plug-ins provide possible actions to be presented to the user based upon the type label associated with the string. The action DLL 215 determines what type label the semantic category includes and cross-references the type label in the type-action database 227 with a list of actions to determine what actions to present to the user. It should be understood that, in a preferred embodiment, the type-action database is not used. Instead, the list of actions is dynamically generated for each type by looking in the registry to determine which actions are installed and then querying the action DLLs to determine which types they apply to.

[0026] After the user chooses an action, the action DLL 215 manages the appropriate action plug-ins 225 and passes the necessary information between the action plug-ins and the application program module 36 so that the action plug-in may execute the desired action. Typically, the application program module sends the action DLL an automation request to invoke the action the user has selected.

[0027] As described above, the combination of the recognized string, type label, metadata and download URL is referred to herein as a semantic category. The type label is a semantic information label. The semantic category may also comprise metadata, which are hidden properties of the semantic category. An example of a semantic category may clarify the definition. Suppose a user enters the text "Gone With the Wind" into an electronic document. The string "Gone With the Wind" may be identified as a semantic category of type label "Book Title" and of type label "Movie Title". In addition, metadata such as the ISBN number may be returned by the recognizer plug-in to the application program module as part of the semantic category. A download URL may be provided with the type labels "Book Title" and "Movie Title" in case the user's machine has not stored action plug-ins for these type labels. For example, an action for the type label "Book Title" may be "Buy this Book" from an online retailer. If the user does not have the action plug-in DLL 225 corresponding to "Buy this book", then the download URL may be used to navigate the user's web browser to an appropriate website to download this action plug-in. In other implementations of the invention, multiple download URLs may be provided for a single type label.

[0028] It should also be understood that the present invention, in a preferred embodiment, also recognizes sequences of capitalized words that contain function words, and which are likely to be special, but for which there is no type label information. These strings are typically labeled by a grammar checker program module.

[0029] The actions provided for a semantic category may utilize both the type label and the text of the recognized string. For example, a word processor program module may use a grammar checker as a recognizer plug-in to label strings that are person names. After a string has been labeled as a person's name, the word processor program module may, through a standard user interface mechanism, allow users to execute pertinent actions, such as looking up the person's name in the contacts folder in a personal information manager program module, sending electronic mail, or searching for the person's name in an HR database.

[0030] Having described an exemplary architecture, an exemplary method 300 for semantically labeling strings during document creation will be described below in reference to Figs. 2 and 3.

Method for Semantically Labeling Strings During Document Creation

[0031] Fig. 3 is a flow chart illustrating a method 300 for semantically labeling strings during creation of an electronic document. Those skilled in the art will appreciate that this is a computer-implemented process that is carried out by the computer in response to input from the user and instructions provided by a program module.

[0032] Referring to Fig. 3, the method 300 begins at start step 305 and proceeds to step 310 when a user opens an electronic document in application program module 36. In a preferred embodiment, the electronic document is a word processing document or a spreadsheet document. However, the invention is not limited to either of these specific types of electronic documents.

[0033] At step 310, the application program module 36 receives a new string, such as when the user enters text, for example a new paragraph, into the electronic document or edits a previously entered paragraph. The method 300 then proceeds to step 315.

[0034] At step 315, the paragraph containing the new string is passed from the application program module 36 to the recognizer DLL 210. The recognizer DLL is responsible for communicating with the application program module, managing the jobs that need to be performed by the recognizer plug-ins, receiving results from the recognizer plug-ins and sending semantic category information to the application program module. At boot time, the recognizer DLL communicates with its recognizer plug-ins to determine what languages it supports, what types it can apply, etc. It should be understood that, in a preferred embodiment, a paragraph is passed to the recognizer DLL at step 315. However, in alternative embodiments, a sentence, the contents of a spreadsheet cell, a section of the document, the entire document, etc. may be passed to the recognizer DLL. In other words, the present invention is not limited to simply passing a paragraph to the recognizer DLL. The method 300 then proceeds to step 320.

[0035] Still referring to step 315, the application program module 36 typically sends one paragraph at a time to the recognizer DLL. In addition, in a preferred embodiment, a grammar checker program module sends all semantic categories (without type labels) to the recognizer DLL that have been identified by the grammar checker program module. Passing these semantic categories (without type labels) to the recognizer DLL is important because doing so saves each recognizer plug-in from needing to decide whether something is a capitalized string interspersed with function words (a task that would require writing a number of regular expressions: Cap Cap Unc Cap; Cap Unc Cap; etc.). If a label is applied by a recognizer plug-in to a string the grammar checker program module labeled, the grammar checker label will then be removed.

[0036] At step 320, during idle time, the paragraph (and information from the grammar checker program module) is passed to the recognizer plug-ins. The method then proceeds to step 325.

[0037] It should be understood that, in a preferred embodiment, the recognizer DLL 210 maintains a job queue. If before the recognizer DLL 210 sends the paragraph to the recognizer plug-ins 220 the user edits the paragraph, then the job containing the edited paragraph is deleted and is not sent to the recognizer plug-ins. Then, a new job enters the queue at step 315 after the edited paragraph is received at step 310. This job deletion is necessary to prevent the recognizer plug-ins from performing unnecessary work on a paragraph that has been edited.

[0038] At step 325, the recognizer plug-ins are executed on the text to search for special strings. For example the recognizer plug-ins are executed on the paragraph to recognize keywords or perform other actions defined by the recognizer plug-in. As part of executing the recognizer plug-in, the paragraph may be broken into sentences by the recognizer plug-in. However, each recognizer plug-in is responsible for its own sentence-breaking. After the keywords are found at step 325, then the method proceeds to step 330.

[0039] At step 330, the results from each of the recognizer plug-ins are received by the recognizer DLL. The method then proceeds to decision step 335.

[0040] At decision step 335, it is determined whether the paragraph that has been reviewed by the recognizer plug-ins has been edited after the paragraph was sent to the recognizer DLL. If so, then the method 300 returns to step 315 and the edited paragraph is received by the recognizer DLL from the application program module. If not, then the method proceeds to step 340.

[0041] At step 340, the results from the recognizer plug-ins are compiled into semantic categories by the recognizer DLL and the semantic categories are sent to the application program module. At step 345, the application program module displays the semantic categories to the user in the electronic document. The method 300 then ends at step 399.

[0042] As should be understood from the above description, the architecture for recognizing semantic categories permits third parties to develop recognizer plug-ins to identify strings of one or more particular types. The recognizer plug-ins communicate with the application program module and receive a string from the application program module. The recognizer plug-ins may apply recognition algorithms to the string and communicate the identity of recognized strings back to the application program module.

[0043] After a string is labeled with a particular type label, the user will be able to execute action plug-ins that pertain to that type label. The action plug-ins preferably are COM objects that are executed via communication between the application program module and the action DLL. Parameters necessary to execute the action (the XML of the string labeled as being of a particular type, the XML of the string representing the current selection) will be passed from the application program module to the action DLL and, in turn, passed to the action plug-in.

Actions Assigned to Type Labels

[0044] An architecture for identifying and executing a set of actions associated with a semantic category may also be provided. This architecture comprises actions that apply to a particular type label (e.g. an action for book titles may be "Buy this book from shop.Microsoft.com") and executing those actions when the user so desires. An action is a user-initiated function applied to a typed string. For example, adding a name to the contacts folder is one action possible for a type label "Person name".

[0045] There is power and flexibility that results from allowing third party vendors, such as IT professionals, to design and write recognizer plug-ins and action plug-ins for deployment within an organization or for deployment on the World Wide Web. Some example actions that may be executed include:

Schedule a meeting
Create task
Display calendar
Add to contacts folder

[0046] Look up in contacts folder, address book, Windows Address Book (WAB), Global

Address List (GAL), etc.
Insert address into document
Send mail to
Display EXPEDIA map
Stock quote lookup
Send instant message to

[0047] Different actions may be assigned to different type labels and these type label-action assignments may be stored in the type-action database 227. Table 1 below illustrates some possible type label-action pairings.

Table 1

Type Labels	Actions
Person name	Show contact info Add to contacts E-mail Insert address into document Send instant message to
Date	Show calendar for that day New task with that due date Schedule meeting that day

Table 1 (continued)

Type Labels	Actions
Place	Display EXPEDIA map Add to contacts
Address	Add to contacts
Phone number	Add to contacts
E-mail	Add to contacts
Date	Schedule a meeting
Task	Schedule a task
Meeting	Schedule a meeting

[0048] For each type label, the type-action database 227 may store a download URL specified by the creator of the type label that users who do not have action-plug-ins or recognizer plug-ins for that semantic category type can go to in order to get action plug-ins and/or recognizer plug-ins. For example, the download URL for the type label "Book Title" might be microsoft.com/semanticcategories.asp. Once at that web page, a user may be offered downloads of various action plug-ins and recognizer plug-ins. There may also be an option on the user interface to navigate to the download URL so that recipients of documents with semantic categories can easily get the action plug-ins for those semantic categories.

Storing Semantic Categories

[0049] Semantic categories may be stored as part of the electronic document along with other document information and may be available when a document is transmitted from one computer to another computer. In a preferred embodiment, storing semantic categories in an electronic document is controlled by an "Embed semantic categories" checkbox. The checkbox is on by default. Turning it off will prevent semantic categories in the document from being saved. The state of the checkbox is per document. The same checkbox controls saving for both .htm and .doc documents.

[0050] Checking a "Save semantic categories as XML properties" checkbox (off by default) will write out the text of all of the semantic categories in the document and their labels in the header of the HTML file in XML (that is using the same tags as are used inline, but surrounded by <xm1> And </xm1>) for easy identification and parsing by search engines and knowledge management systems.

[0051] Semantic categories may be saved as a unique namespace plus a tag name. A namespace is an XML construct for uniquely identifying a group of XML tags that belong to a logical category. Thus, every semantic category is uniquely identified by its nametag (e.g., "streetname") in addition to its namespace (e.g., "schemasmicrosoft-com: outlook:contact")

[0052] Although the method 300 described above is one method for identifying semantic categories, there may be other mechanisms for identifying semantic categories. One mechanism is a grammar checker program module (not shown) connected to word processor program module 37. Another mechanism is receiving a semantic category from another electronic document. For example, when text containing a semantic category is copied from one electronic document and passed into another electronic document of the word processor program module 37, the information identifying the semantic category is preserved and copied along with the copied text.

Displaying Semantic categories to the User

[0053] Referring now to Fig. 4, an illustration of a display of a semantic category 400 and its associated dropdown menu 405 will be described. It should be understood that Fig. 4 is an illustration of a semantic category 400 and dropdown menu 405 as displayed to a user by the application program module 36.

[0054] The string 410 associated with semantic category 400 is the string "Bob Smith". As shown in Fig. 4, the string 410 of a semantic category 400 may be identified to the user by brackets 415. Of course, many other devices such as coloring, underlining, icons, etc. may be used to indicate to the user that a particular string is a semantic category.

[0055] In a preferred embodiment, when the user hovers a cursor over the string 410 or places the insertion point within string 410, then dropdown menu 405 is displayed to the user. The dropdown menu may display a list of actions associated with a semantic category. The dropdown menu may appear above and to the left of the semantic category string.

[0056] Typically, the first line of the dropdown menu indicates which string is the semantic category string (Bob Smith in Fig. 4) and what type the semantic category is (Person name in Fig. 4). Listed below the first line are actions 420 available for the semantic category type, such as "Send mail to...", "Insert Address", and "Display contact information...".

[0057] The first item on the drop down menu below the separator line is "Check for new actions..." 425. "Check for new actions..." 425 will appear only for semantic categories whose download URL is available to the application program module. If selected, "Check for new actions..." 425 uses the semantic category download URL to navigate the user's web browser to the homepage for the semantic category type applied to the string. For example, suppose new actions have been defined for the semantic category type "person name". If so, then new actions will be downloaded to the user's computer after selecting "Check for new actions..." 425. "Check for new actions..." 425 will be grayed out if a download URL is unavailable for the semantic category.

[0058] If selected, the "Remove this semantic category" item 430 deletes the semantic category label from the string. If selected, the "Semantic categories" item 435 navigates the user to the semantic categories tab of the autocorrect dialog.

[0059] It should be understood that the application program module sends a request to the action DLL to determine which actions are shown with each semantic category type.

Actions Performed in Association with Semantic categories

[0060] There are a number of functions that users perform on typed data that preferred word processor program module 37 and semantic categories will make easier. The functions fall into three primary categories:

1) interacting with personal information manager contacts, tasks, meetings, and mail;
2) interacting with properties on the World Wide Web or a corporate intranet; and
3) interacting with other applications on the client machine.

[0061] A single string may be associated with multiple semantic categories. Every semantic category has a type label with one or more action plug-ins defined for the type label. For example, the "Address" type label may have the "Open in Mappoint", "Find with Expedia Maps" and "Add to my Address Book" actions associated with it and each of these actions may have a different action plug-in to execute the action.

[0062] The actions assigned to type labels also depend on the computer that the application program module is running on. Thus, if a computer has three actions registered for the type label "Address", then all strings with an "Address" type label will be assigned to three actions. However, if one of these semantic categories is sent to a computer which has only two actions registered for the "Address" type label, then the user will only be exposed to two actions for this semantic category.

Nesting of Semantic categories

[0063] In an embodiment of the present invention, semantic categories may be nested inside each other. For example, the string "George Washington" may include a semantic category with type label "Person Name" for the span "George Washington State" and a semantic category with type label "State" for the span "Washington". Moreover, two semantic categories may cover exactly the same span. For example, the string "George Washington" may include a semantic category with type label "Person Name" and a semantic category with type label "President".

[0064] Because the preferred application program module 37 will support labeling a single string with multiple type labels (e.g. Bob Smith could be a semantic category labeled as a "Person Name" and labeled as a "Microsoft employee"), the preferred application program module 37 will use cascade menus on the dropdown menu if multiple semantic category types are assigned.

[0065] For example, the cascade menu may include a list of the type labels included in the recognized string. This list may include a type label "Person Name" and a type label "Microsoft employee".

[0066] It should be understood that a cascade menu may be used to allow the user to select which type label the user is interested in and to further select an action after selecting the type label.

In-document User Interface to Indicate Semantic categories

[0067] As described above with reference to Fig. 4, the application program module may include the option to display an in-document user interface to indicate the location of semantic categories. This in-document user interface may use a colored indication to indicate the location of a semantic category, such as the brackets 415 in Fig. 4. The in-document user interface will also be able to show nesting of semantic categories. For example, if Michael Jordan is labeled as a semantic category with type label "Person Name", Michael is a semantic category with type label "First Name" and Jordan is a semantic category with type label "Last Name", the document may look like this with the brackets indicating semantic categories:

[[Michael][Jordan]]

[0068] Of course, the in-document user interface may be any sort of indication. For example, in the "EXCEL" spreadsheet application program, the interface comprises a triangle in the lower right hand portion of a cell to indicate that one or more semantic categories are present in the cell.

[0069] Although the present invention has been described as implemented in a word processing program module, it should be understood that the present invention may be implemented in other program modules, including, but not limited to, HTML authoring programs and programs such as the "POWERPOINT"® presentation graphics program and the "OFFICE" program module, both marketed by Microsoft Corporation of Redmond, Washington.

[0070] As described above, the semantic category may also include metadata returned by the recognizer plug-ins. For example, a recognizer plug-in that recognizes the titles of books may return as metadata an ISBN book number when it recognizes the title of a book. The ISBN book number metadata may then be used to provide actions. Metadata may also be used to disambiguate for actions and searches. For example, suppose a recognizer DLL is linked to a corporate employee database to recognize names. When the recognizer DLL recognizes "Bob Smith", it may store "employeeID=12345" as metadata in the background. Then, when an action is fired, the text in question will be known to reference Bob Smith, employee no. 12345 rather than Bob Smith, employee no. 45678. Also, the metadata may allow searches to be performed independent of the actual text in a document. So, a search may be conducted on "Robert Smith" by looking for employee 12345 in the employee databases and by performing a search on the metadata for employee number 12345 to find documents with "Bob Smith" in them. There are also numerous other functions for metadata. For instance, DHTML could be inserted so special features of a web browser, such as text color or font size, may be used. Additionally, data used by other actions may be inserted such as someone's e-mail address that could be used by the send-mail-to action, a normalized version of the date could be stored to easily interact with a personal information manager, etc.

Implementation of Exemplary Application Program Interfaces

[0071] According to an exemplary embodiment, application program interfaces (API) are provided for implementing the recognizer plug-ins 220 and the action plug-ins 225, discussed above. Referring to Figs. 2 and 5, the recognizer API 250 includes a group of properties and methods or instructions that allow the recognizer plug-ins 220 to semantically label strings of text or cells of information, as discussed with reference to Figs. 2 and 3. The action API 250 includes properties and methods or instructions that are called upon when a user initiates a particular action, for example, opening a Web browser, going to a particular URL, or opening an instance of a word processing application, a spreadsheet application, and the like.

[0072] Referring to Fig. 5, the properties and methods of the recognizer API 250 are illustrated. The Description property 251 includes a description of the recognizer plug-ins 220. The Name property 252 includes a name for the recognizer plug-in as it appears in a user interface such as a tools menu or options contained in a dialog box of an exemplary application program. The Prog(programmatic)ID 253 includes a unique identifier for the object class of the recognizer DLL. The Recognize method 254 includes instructions and routines for recognizing character strings as actionable after having been semantically labeled, as described above. The LabelCount property 236 describes the number of semantic label types the associated recognizer DLL recognizes, for example, book titles, movie titles, etc. The DownloadURL property 256 is the URL that is navigated to download additional semantic label types or categories. The LabelName property 257 includes unique identifiers for the semantic label types that the recognizer DLL supports.

[0073] The Description property 231 of the action API 230 includes a description of the semantic label action associated with a given action, for example, opening a Web browser. The InvokeVerb method 232 includes the routines and instructions that are executed when an action item is initiated, as described above. The Name property 233 includes a title for the given action. The ProgID property 234 includes a unique identifier of the object class for the particular action. The LabelCaption property 235 includes the caption that will be displayed at the top of a user interface such as an actions menu. The LabelCount property 236 includes the number of semantic label types recognized by a corresponding recognizer DLL or corresponding recognizer plug-ins. The LabelName property 237 identifies the types of semantic label actions, for example, opening a Web browser, navigating to a particular URL, opening an instance of an application program, etc.

[0074] The VerbCaptionFromID property 238 includes the captions for available actions provided in a user interface, such as an actions menu. The VerbCount property 239 includes how many actions are supported for a given semantic label type. The VerbID property 240 returns a unique identifier within the semantic label for use within a given application program. This mechanism is supplied so that the semantic label recognizer DLL and recognizer plug-ins can mix and match semantic label actions for various semantic label types supported by the DLLs and plug-ins. The VerbNameFromID property 241 returns a name to represent the semantic label action for use internally within a given application program. For example, for the semantic label action string "view company Website", the VerbNameFromID property

241 may return a name such as "viewCompanyWebsite".

[0075] The following is a discussion of an exemplary implementation of the recognizer API 250 and the action API 230. Following a discussion of various components of these APIs is exemplary code written in Microsoft Visual Basic 6.0 for a recognizer DLL and action DLL for locating a "Fourth Coffee" flavor in a set of coffee flavors. The code and the example are exemplary only and are not limiting of the scope of the invention described herein.

Implementing the Recognizer API

[0076] To implement the recognizer API 250, the ProgID property 253, the Name property 252, and the Description property 251 are populated by specifying the name and by providing a description of the particular recognizer DLL. Additionally, a locale identifier is passed to the recognizer DLL, for example, Spanish, German, English, etc., to give identification of the user interface language in which semantic labeling will occur for the particular recognizer DLL. The following is exemplary code written in Microsoft Visual Basic 6.0 for populating the ProgID property 253, the Name property 252, and the Description property 251.

```
Private Property Get LabelRecognizer_ProgId()
      As String


   LabelRecognizer_ProgId =
      "CoffeeFlavor.LabelRecognizer"
End Property

Private Property Get LabelRecognizer_Name
      (ByVal LocaleID As Long) As String
   LabelRecognizer_Name =
      "Coffee Flavors"
End Property

Private Property Get LabelRecognizer_Desc
      (ByVal LocaleID As Long) As String
   LabelRecognizer_Desc =
      "Directs users Coffee Flavors"
End Property
```

[0077] Next, the number of semantic label types, the list of semantic label action types, and the additional semantic labels download location of the recognizer DLL are provided by populating the LabelCount property 255, the LabelName property 257 and the DownloadURL property 256. The number of semantic label types supported by the recognizer DLL 210 are populated. For example, if a value of two (2) for the LabelCount property is returned, there will be two subsequent calls made to the LabelName property with a label ID value of one(1) passed in the first call and a value of two passed in the second call. The download location includes the HTTP address used (download URL) to download tools associated with particular semantic labels. If no download URL is available, the value of the property is set to null. The following is exemplary code for populating the LabelCount property 255, the LabelName property 257 and the DownloadURL property 256.

```
Private Property Get LabelRecognizer_LabelCount()
    As Long
  LabelRecognizer_LabelCount = 1


End Property


Private Property Get LabelRecognizer_LabelName
    (ByVal LabelID As Long) As String
  If LabelID = 1 Then
    LabelRecognizer_LabelName =
      "schemas-fourth-com/fourthcoffee#flavor"
  End If
End Property


Private Property Get LabelRecognizer_DownloadURL
    (ByVal LabelID As Long) As String
  LabelRecognizer_DownloadURL = Null
End Property
```

[0078] A list of items such as text in a word processing application or information contained in a cell of a spreadsheet is provided to the Recognize method 254. The text or information is provided to the recognizer plug-ins 220 of the recognizer DLL as a text string and the form of the text, for example, paragraph, cell, etc., is provided. The following is exemplary programming code for building a list of strings associated with, for example, coffee flavors to be recognized by the recognizer plug-ins 220 of the recognizer DLL 210. According to an exemplary embodiment, to avoid rebuilding the list of strings every time there is a call to recognize a text item, the list is built and populated in an array when the class of semantic label types is created.

```
Dim garrTerms(6) As String
Dim gintNumTerms As Integer

Private Sub Class_Initialize()
   garrTerms(1) = "latte"
   garrTerms(2) = "carmelito"

  garrTerms(3) = "verona"
  garrTerms(4) = "Columbia blend"
  garrTerms(5) = "antigua"
  garrTerms(6) = "kona"
  gintNumTerms = 6
End Sub
```

[0079] After the list of strings to be recognized is built, as described above, the Recognize method 254 is constructed that will search for strings in the array created above. The search performs a case-insensitive search through the passed-in strings of text for each of the terms supplied. For the example given above, the strings supplied include a variety of coffee flavors. Construction of the Recognize method 254 includes passing in the language identifier of the text strings to be searched by the recognizer DLL, for example, German, English, etc. Additionally, an object is passed in to add additional semantic labels. A Property Bag property allows for the addition of new attributes/value pairs to be added to semantic labels. For example, the name "Bob Smith" may be set up as a name type, but the Property Bag property allows for the storage of other values like employee ID, telephone number, address by adding new attribute/value pairs to the semantic label type. For example, the attribute "employee ID" and the value "123" may be added to the name type for "Bob Smith." Information is also passed in to define the name space in the document, and to specify the start position of the string and the length of the string. The following is exemplary programming code for constructing the Recognize method 254.

```
Private Sub LabelRecognizer_Recognizer _Recognize
     (ByVal Text As String,
     ByVal DataType As LabelLib.IF_TYPE,
     ByVal LocaleID As Long,
     ByVal RecognizerSite As
        LabelLib.LabelRecognizerSite)
  Dim intLoop As Integer
```

```
        Dim intIndex As Integer
        Dim intTermLen As Integer
        Dim stlPropertyBag As LabelLib.ILabelProperties
        Text = LCase(String:=Text)
        For intLoop = 1 To gintNumTerms
           intIndex = InStr(Text, garrTerms(intLoop))
           intTermLen = Len(garrTerms(intLoop))
           Do While intIndex > 0
              Set stlPropertyBag =
                 RecognizerSite.GetNewPropertyBag
              RecognizerSite.CommitLabel
                 "schemas-fourth-com/fourthcoffee#flavor", intIndex,
                 intTermLen, stlPropertyBag
              intIndex = InStr(intIndex + intTermLen,
                 Text, garrTerms(intLoop))
           Loop
        Next intLoop
     End Sub
```

Implementing the Action API

[0080] After construction of the Recognize method 254, the action API 230 is constructed by first populating the ProgID property 234, the Name property 233, and the Description property 231. Population of those properties is similar to population of similar properties for the recognizer API 250, described above. The following is exemplary programming code for populating the ProgID property 234, the Name property 233, and the Description property 231 of the action API 230.

```
        Private Property Get LabelAction_ProgId() As String
           LabelAction_ProgId =
              "CoffeeFlavor.LabelAction"
        End Property
```

```
Private Property Get LabelAction_Name
    (ByVal LocaleID As Long) As String
  LabelAction_Name = "Coffee Flavor actions"
End Property

Private Property Get LabelAction_Desc
    (ByVal LocaleID As Long) As String
  LabelAction_Desc =
    "Provides actions for certain Coffee Flavors"
End Property
```

[0081] The recognizer DLL 210 is informed of the number, names, and captions of the semantic label types by populating the LabelCount property 236, the LabelName property 237, and the LabelCaption property 235. For each label type supported by the recognizer plug-ins 220 of the recognizer DLL 210, the recognizer DLL 210 is informed of the number of actions. If there are three actions for the label type "persons" and two actions for the label type "companies," then a total of five label types are passed to the recognizer DLL. A name for each action class is provided and a caption that will appear in an appropriate user interface such as an action menu is provided. For example, the caption "Coffee flavors at Joe's Coffee House" might be provided for an action that will navigate to the Web page of Joe's Coffee House for a semantically labeled coffee flavor. In addition, according to an exemplary embodiment, the application programs applicable to a given action may be identified so that inapplicable application programs may be disabled. For example, if the semantically labeled information is not applicable for use in a spreadsheet, a spreadsheet application may be disabled. The following is exemplary programming code for populating the LabelCount, the LabelName and LabelCaption properties.

```
Private Property Get LabelAction_LabelCount()
    As Long
  LabelAction_LabelCount = 1
End Property
```

```
Private Property Get LabelAction_LabelName
    (ByVal LabelID As Long) As String
  If LabelID = 1 Then
    LabelAction_LabelName =
      " schemas-fourth-com/fourthcoffee#flavor "
  End If
End Property

Private Property Get LabelAction_LabelCaption
    (ByVal LabelID As Long,
    ByVal LocaleID As Long)
    As String
  LabelAction_LabelCaption =
    "Selected Coffee Flavors"
End Property
```

[0082] Next, the recognizer DLL is informed of the number, names, and identifiers of supported semantic label actions by populating the VerbCount property 239, the VerbID property 240, the VerbCaptionFromID property 238, and the VerbNameFromID property 241. Population of these properties is done on a label type by label type basis to tell the label actions about verbs that are supported. Generating a unique ID for a particular verb is performed by the action DLL 215 which gives the action DLL more flexibility. For example, a semantic label action DLL can specify the same VerbID value for the same action across varying semantic label types, or the DLL can use the same VerbID for similar variants of an action. The following is exemplary programming code for populating the VerbCount, VerbID, VerbCaptionFromID, and VerbNameFromID properties.

```
Private Property Get LabelAction_VerbCount
    (ByVal LabelName As String) As Long
  If LabelName = " schemas-fourth-com/fourthcoffee#flavor " Then
```

```
        LabelAction_VerbCount = 6
    End If
End Property

Private Property Get LabelAction_VerbID
        (ByVal LabelName As String,
        ByVal VerbIndex As Long)
        As Long
    LabelAction_VerbID = VerbIndex
End Property

Private Property Get LabelAction_VerbCaptionFromID
        (ByVal VerbID As Long,
        ByVal ApplicationName As String,
        ByVal LocaleID As Long)
        As String
    Select Case VerbID
        Case 1
            LabelAction_VerbCaptionFromID =
                "latte"
        Case 2
            LabelAction_VerbCaptionFromID =
                "carmelito"
        Case 3
            LabelAction_VerbCaptionFromID =
                "verona"
        Case 4
            LabelAction_VerbCaptionFromID =
                "Columbia blend"
        Case 5
            LabelAction_VerbCaptionFromID =
                "antigua"
            Case 6
                LabelAction_VerbCaptionFromID =
                    "kona"
        End Select
    End Property

    Private Property Get LabelAction_VerbNameFromID
            (ByVal VerbID As Long) As String
        Select Case VerbID
            Case 1
                LabelAction_VerbNameFromID = "latte"
            Case 2
                LabelAction_VerbNameFromID = "carmelito"
            Case 3
                LabelAction_VerbNameFromID = "verona"
            Case 4
                LabelAction_VerbNameFromID = "Columbia blend"
            Case 5
                LabelAction_VerbNameFromID = "antigua"
            Case 6
                LabelAction_VerbNameFromID = "kona"
        End Select
    End Property
```

[0083] After those properties are populated, the InvokeVerb method 232 is constructed for invoking the appropriate verb when the user selects an action displayed in the semantic label user interface corresponding to the VerbCaptionFromID value. According to the exemplary programming code provided below, six actions are provided which are hyperlinks that navigate to six coffee flavor Web sites. Construction of the InvokeVerb method includes identifying the verb, identifying the action, and naming the application program 36 so that different actions may be performed depending on the context, for example, word processing text versus cell data in a spreadsheet.

[0084] According to an exemplary embodiment, a pointer may be given to an application program's 36 object model so that the action DLL may use the object model to manipulate text in a document, insert text, manipulate data in a spreadsheet, etc. The label text, the label type, and contents of a property bag are provided, and that information may be provided according to alternate computing languages, for example XML, so that the information may be manipulated using alternate language systems, such as XML parsers. This aspect is useful for manipulating multiple nested semantic labels, for example, street, city, state, and zip code. The following is exemplary programming code for constructing the

InvokeVerb method 232 of the action API 230.

```
Private Sub LabelAction_InvokeVerb
    (ByVal VerbID As Long,
    ByVal ApplicationName As String,
    ByVal Target As Object,
    ByVal Properties As LabelLib.ILabelProperties,
    ByVal Text As String,
    ByVal Xml As String)
  Dim ieInternetExplorer As Variant
  Set ieInternetExplorer =
    CreateObject("InternetExplorer.Application")
  With ieInternetExplorer
    Select Case VerbID
      Case 1
        .Navigate2 "www.latte.com"
      Case 2
        .Navigate2 "www.carmelito.com"
      Case 3
        .Navigate2 "www.verona.com"
      Case 4
        .Navigate2 "www.columbiablend.com"


            Case 5
              .Navigate2 "www.antigua.com"
            Case 6
              .Navigate2 "www.kona.com"
          End Select
          .Visible = True
        End With
      End Sub
```

Registration of Application Programming Interfaces

[0085] In order for the application program module, such as a word processing application or a spreadsheet application, to know that the recognizer and action plug-ins and DLLs constructed and made operable by the construction of the APIs, discussed above, are actually plug-ins and DLLs for use by the application, the plug-ins and DLLs must be registered in the operating system registry of the user's computer 20. Accordingly, information is written into the registry of the operating system 35 so that any plug-ins and DLLs referred to by the semantic label properties in that portion of the system registry will be booted by the applicable application program utilizing the semantic labeling functionality when that application programs boots.

[0086] Although the present invention has been described above as implemented in a preferred application program module, it will be understood that alternative embodiments will become apparent to those skilled in the art to which the present invention pertains without departing from its spirit and scope. Accordingly, the scope of the present invention is defined by the appended claims rather than the foregoing description.

Claims

1. In a system for semantically labeling a string of text in an electronic document created in an application program module, a method of implementing an application program interface for further implementing a recognizer plug-in, the method comprising the steps of:

   providing an identification, a title, and a description for the recognizer plug-in;
   providing a number of text label types and a download location for additional label types;
   providing a list of text items for recognition by the recognizer plug-in; and
   causing the recognizer plug-in to search for text items from the list of text items in a selection of text received by the recognizer plug-in.

2. The method of Claim 1, further comprising the step of registering the recognizer plug-in in an operating system registry.

3. The method of Claim 1, wherein the step of providing an identification, a title, and a description for a recognizer plug-in includes the step of providing the name of the recognizer plug-in for presentation via a user interface.

4. The method of Claim 1, wherein the step of providing a number of text label types includes providing a number of text label types recognizable by the recognizer plug-in.

5. The method of Claim 1 whereby the step of providing a download location for additional text label types includes providing a uniform resource locator (URL) for obtaining additional text label types.

6. In a system for semantically labeling a string of text in an electronic document created in an application program module, a method of implementing an application program interface for further implementing an action plug-in, the method comprising the steps of:

   providing an identification, a title, and a description for the action plug-in;
   providing to a recognizer DLL a number of text label types, a name for each of the number of text label types, and a caption for each of the number of text label types;
   providing the recognizer DLL a number of text label actions, a name for each of the number of text label actions, and an identification for each of the number of text label actions; and
   displaying a list of the text label actions upon user initiation.

7. The method of Claim 6, whereby prior to the step of displaying a list of the text label actions upon user initiation, further comprising the step of providing names of each of the number of text label actions for presentation in a user interface.

8. The method of Claim 7, wherein the step of providing to a recognizer DLL a caption for each of the number of text label types, further comprises the step of providing captions for displaying in the user interface each of the number of text label types.

9. The method of Claim 6, wherein the step of providing to a recognizer DLL a number of text label types, includes providing a number of text label types for which the action plug-in provides actions.

10. The method of Claim 6 further comprising the steps of registering the recognizer DLL and the action plug-in in a computer operating system registry.

11. In a system for semantically labeling a string of text in an electronic document created in an application program module, a method of implementing an application program interface for further implementing a recognizer plug-in and an action plug-in, the method comprising the steps of:

providing an identification, a title, and a description for the recognizer plug-in;
providing the recognizer plug-in number of text label types, a list of text label action types, and a download location for additional label types;
providing the recognizer a list of text items for recognition by the recognizer plug-in;
providing to the recognizer plug-in a number of text label types, a name for each of the number of text label types, and a caption for each of the number of text label types;
providing the recognizer plug-in a number of text label actions, a name for each of the number of text label actions, and an identification for each of the number of text label actions;
providing an identification, a title, and a description for the action plug-in;
causing the recognizer plug-in to search for text items from the list of text items in a selection of text received by the recognizer plug-in; and
displaying a list of the text label actions upon user initiation.

12. The method of Claim 11, further comprising the step of registering the recognizer plug-in and the action plug-in in an operating system registry.

13. The method of Claim 11, wherein the step of providing an identification, a title, and a description for a recognizer plug-in includes the step of providing the name of the recognizer plug-in for presentation via a user interface.

14. The method of Claim 11, wherein the step of providing a number of text label types includes providing a number of text label types recognizable by the recognizer plug-in.

15. The method of Claim 11, wherein the step of providing a list of text label action types includes providing a list of text label action types associated with an action plug-in and recognizable by the recognizer plug-in.

16. The method of Claim 11 whereby the step of providing a download location for additional text label types includes providing a uniform resource locator (URL) for obtaining additional text label types.

17. In a system for semantically labeling a string of text in an electronic document created in an application program module, the system including a recognizer plug-in and an application program interface (API) for allowing the recognizer plug-in to semantically label strings of text, comprising:

a description property including a description of the recognizer plug-in;
a name property including a name of the recognizer plug-in;
a programmatic identifier including a unique identifier for the recognizer plug-in;
a label count property identifying a number of semantic label types recognizable by the recognizer plug-in;
a label name property including unique identifiers for the semantic label types recognizable by the recognizer plug-in;
a label download URL property for locating additional semantic label types recognizable by the recognizer plug-in; and
a recognize method including instructions which when executed by a computer cause the recognizer plug-in to recognize character strings semantically labeled as character string types recognizable by the recognizer plug-in.

18. The application program interface of Claim 17, further comprising an action API, comprising
a description property describing a semantic label action associated with an action operable on a semantically labeled text string;
a name property including a title for the semantic label action;

a programmatic identifier property including a unique identifier for the semantic label action;

a label caption property for displaying via a user interface;

a label name property identifying a number of types of semantic label actions operable by the action plug-in;

a label count property identifying the number of semantic label types recognizable by the recognizer plug-in;

a verb caption from ID property including captions for the number of semantic label actions provided via a user interface;

a verb count property including a quantity of semantic label actions supported by a semantic label type;

a verb ID property for returning a unique identifier within a semantic label for use within the application program module;

a verb name from ID property for returning a name to represent a semantic label action for use within the application program module; and

an invoke verb method including instructions which when executed by a computer initiate the semantic label action.

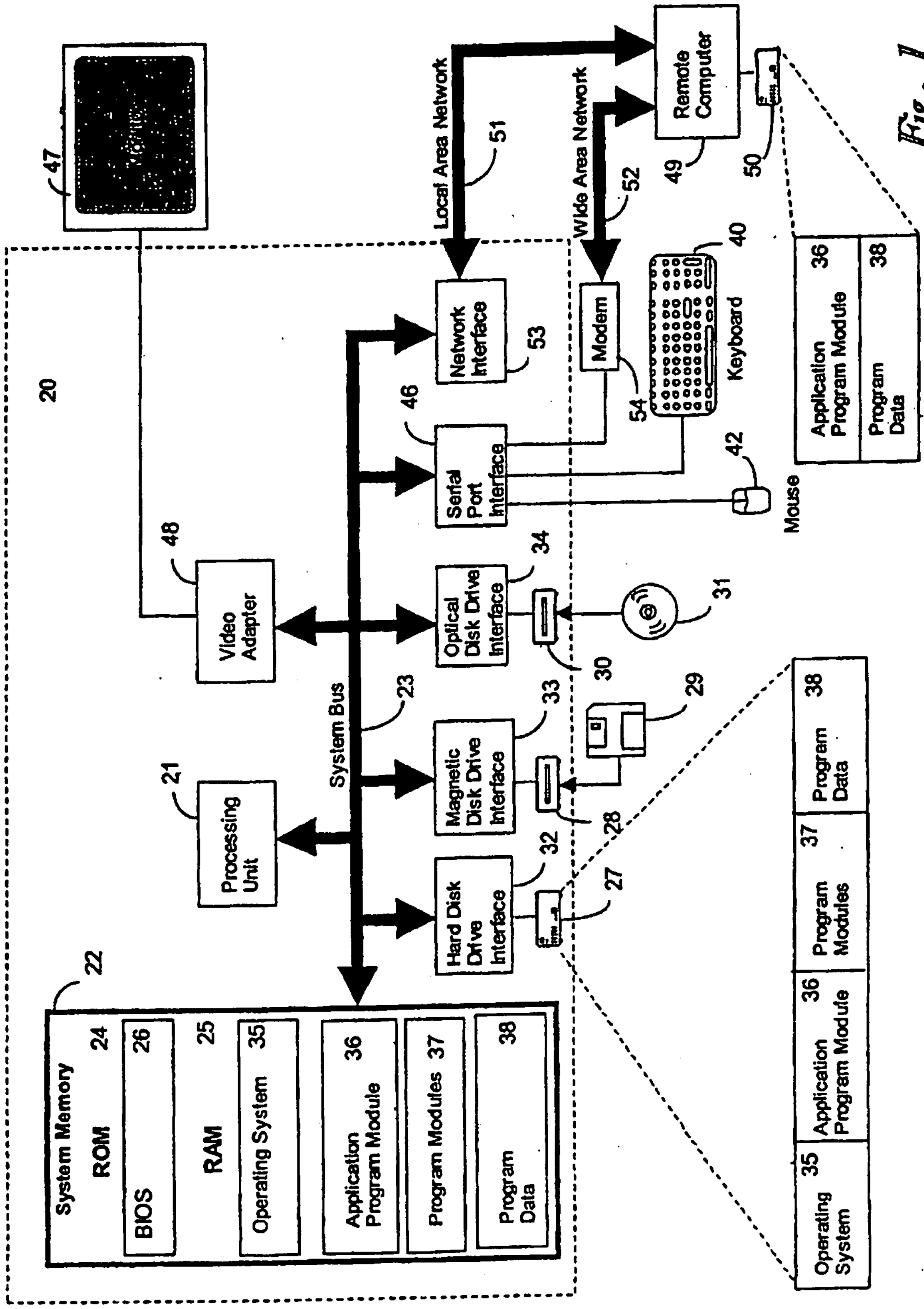
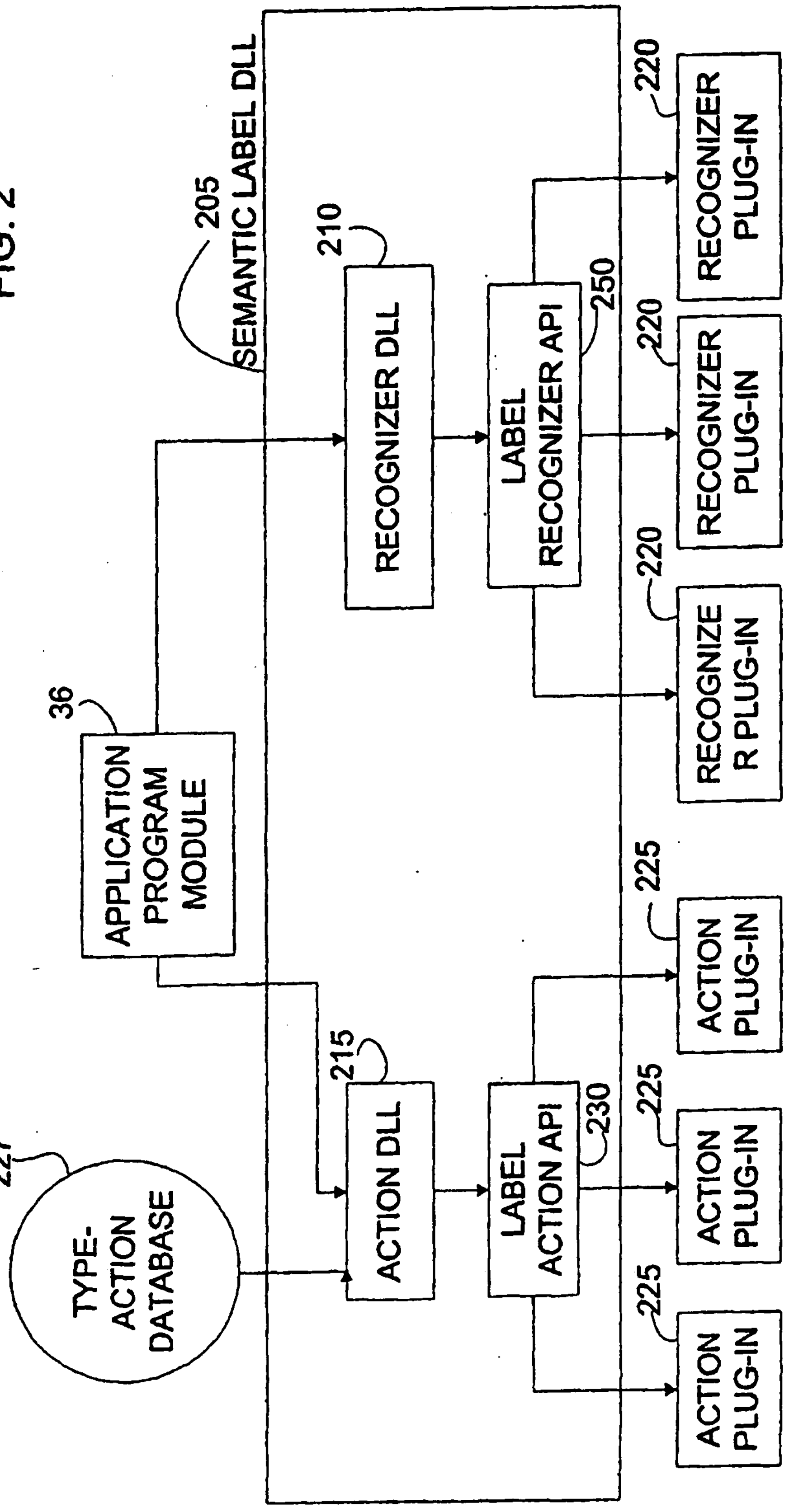

20
System Memory
ROM 24
BIOS 26
RAM 25
Operating System 35
Application Program Module 36
Program Modules 37
Program Data 38
22
Processing Unit
21
System Bus
23
Video Adapter
48
Monitor
47
Hard Disk Drive Interface
32
27
Magnetic Disk Drive Interface
33
28
29
Optical Disk Drive Interface
34
30
31
Serial Port Interface
46
Network Interface
53
Modem
54
Keyboard
40
Mouse
42
Local Area Network
51
Wide Area Network
52
Remote Computer
49
50
Operating System 35
Application Program Module 36
Program Modules 37
Program Data 38
Application Program Module 36
Program Data 38

Fig. 1

FIG. 2


205
SEMANTIC LABEL DLL
210
RECOGNIZER DLL
LABEL RECOGNIZER API
250
220
RECOGNIZER PLUG-IN
220
RECOGNIZER PLUG-IN
220
RECOGNIZE R PLUG-IN
36
APPLICATION PROGRAM MODULE
215
ACTION DLL
LABEL ACTION API
230
225
ACTION PLUG-IN
225
ACTION PLUG-IN
225
ACTION PLUG-IN
227
TYPE-ACTION DATABASE

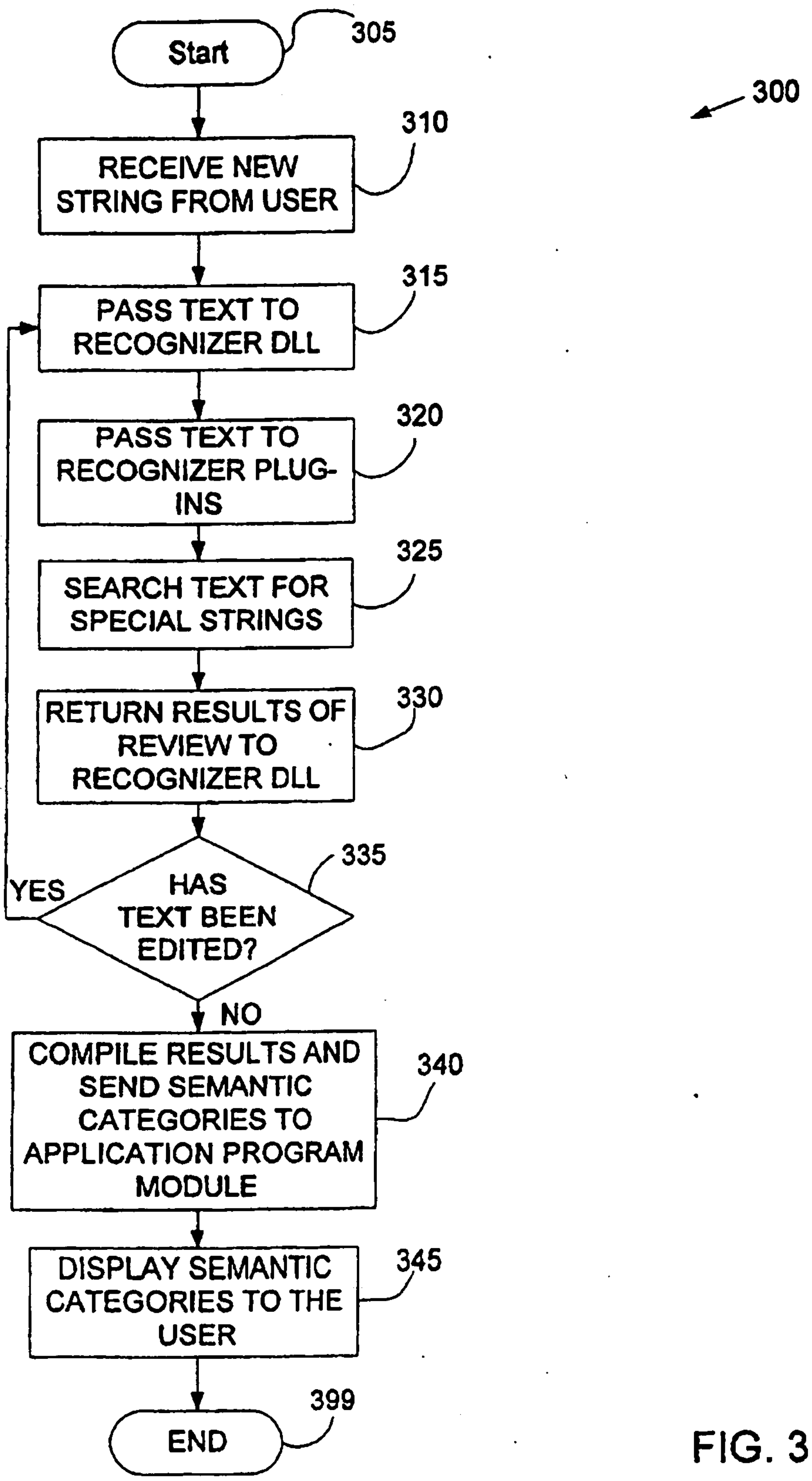

300
Start
305
RECEIVE NEW STRING FROM USER
310
PASS TEXT TO RECOGNIZER DLL
315
PASS TEXT TO RECOGNIZER PLUG-INS
320
SEARCH TEXT FOR SPECIAL STRINGS
325
RETURN RESULTS OF REVIEW TO RECOGNIZER DLL
330
HAS TEXT BEEN EDITED?
335
YES
NO
COMPILE RESULTS AND SEND SEMANTIC CATEGORIES TO APPLICATION PROGRAM MODULE
340
DISPLAY SEMANTIC CATEGORIES TO THE USER
345
END
399

FIG. 3

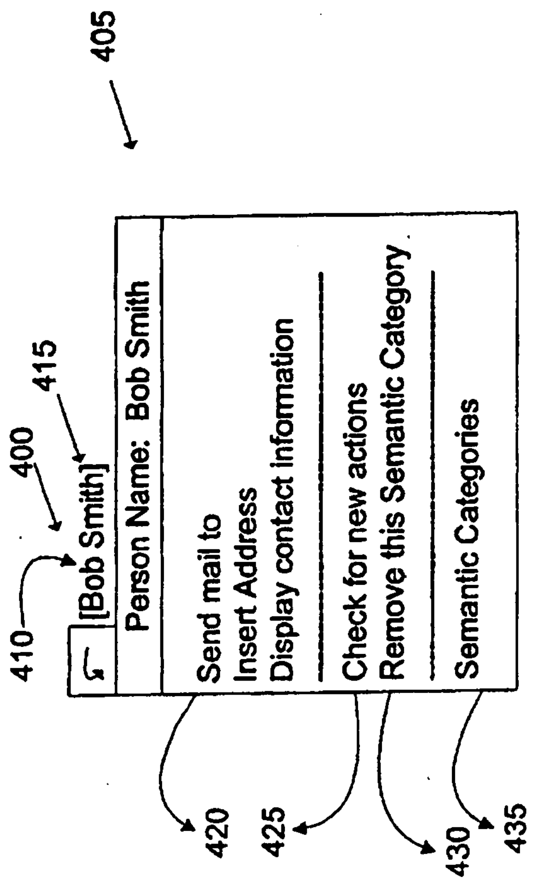

405
400
415
410
[Bob Smith]
Person Name: Bob Smith
Send mail to
Insert Address
Display contact Information
Check for new actions
Remove this Semantic Category
Semantic Categories
420
425
430
435

FIG. 4

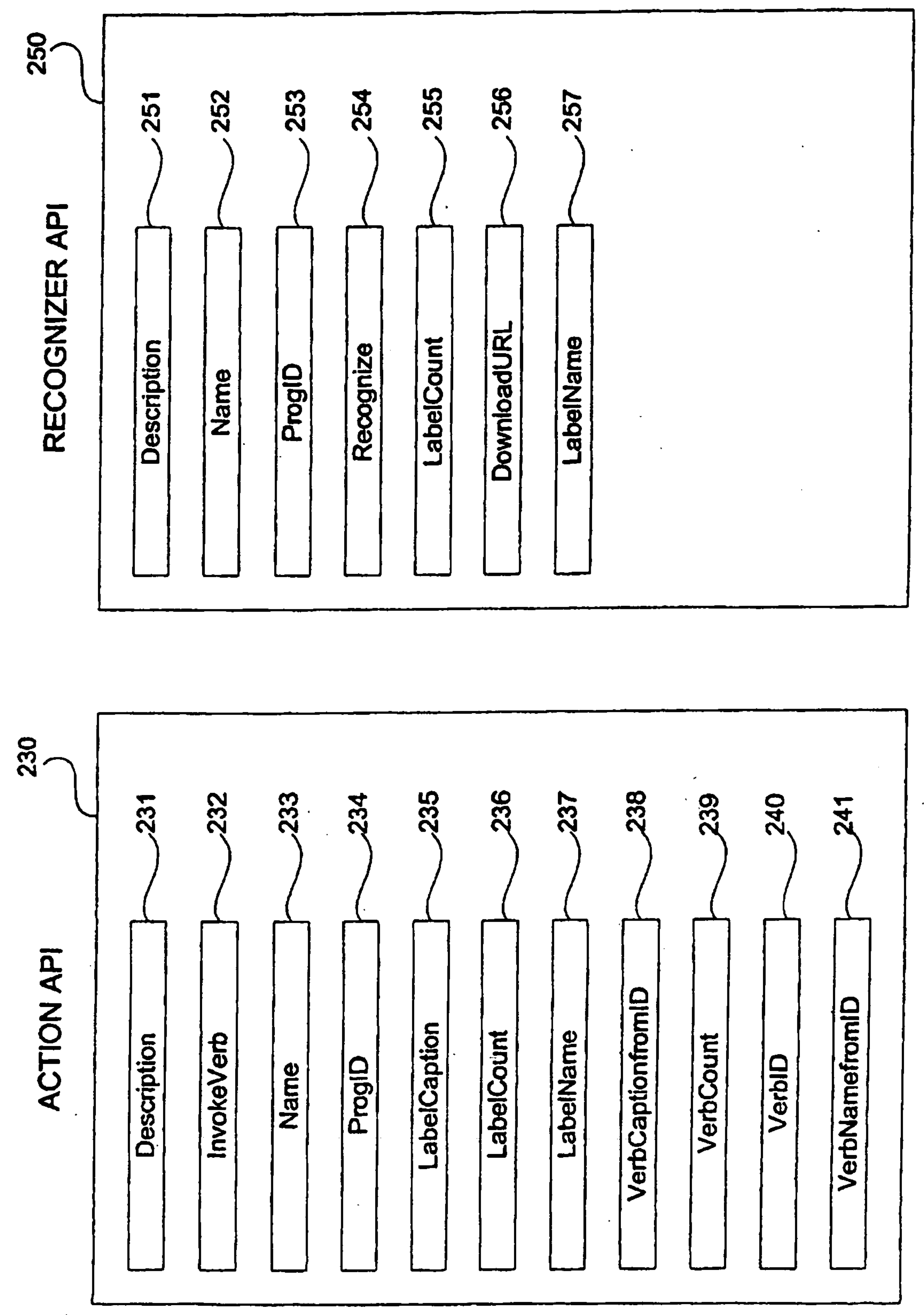

RECOGNIZER API
250
251
252
253
254
255
256
257
Description
Name
ProgID
Recognize
LabelCount
DownloadURL
LabelName
ACTION API
230
231
232
233
234
235
236
237
238
239
240
241
Description
InvokeVerb
Name
ProgID
LabelCaption
LabelCount
LabelName
VerbCaptionfromID
VerbCount
VerbID
VerbNamefromID

FIG. 5